[**Department of Computer Science**](http://www.cs.umd.edu/)

[**CMSC131:**](http://www.cs.umd.edu/class/fall2017/cmsc131-FC01/) Fall 2017

**Project:** Media Rental Manager

**Due Date:** Fri Dec 8, 11:58 pm

## Overview

For this project you will implement a simplified version of a system that allows people to rent movies and music albums (similar to Netflix) and receive them via mail. For this project you will define all the classes needed (yes it is your design :)).

**You must implement this project by yourself. DO NOT POST ANY INFORMATION in Piazza regarding the classes/interfaces you are planning to use in your design. If you post any information you will be violating academic integrity rules.**

This project have been used in previous semesters. Using any implementation you might found online or from another student represents an academic integrity violation.

## Objectives

To practice design, inheritance, ArrayList and javadoc.

## Grading

* (30%) Public Tests
* (10%) Release Tests
* (20%) Inheritance Requirement
* (22%) Design that includes at least three classes / interfaces (not including MediaRentalManager)
* (8%) ArrayList Requirement
* (5%) Javadoc
* (5%) Style (Good Class/Variable/Method Names, Indentation)

## Clarifications

## Code Distribution

The project's code distribution is available by checking out the project named **MediaRentalManager**. The code distribution provides you with the following:

* A package named **mediaRentalManager**  - In this package you will provide the implementation for the system. We have provided an interface( [MediaRentalManagerInt](http://docs.google.com/doc/mediaRentalManager/MediaRentalManagerInt.html) ) that specifies the functionality your system is expected to implement. The complete javadoc for the project can be found at [javadoc](http://docs.google.com/doc/index.html).
* A package named **tests**  - Includes the public tests and the shell file for your student tests.

## Specifications

### What You Must Implement

You must define a class named **MediaRentalManager** (we did not provide it) that implements the MediaRentalManagerInt interface functionality. You must define classes that support the functionality specified by the interface. The following specifications are associated with the project:

1. You should study the public tests and the output (files with .txt extension) to familiarize yourself with the expected system functionality.
2. Define a class named **MediaRentalManager**. Feel free to add any instance variables you understand are needed or any private methods. Do not add any public methods (beyond the ones specified in the MediaRentalManagerInt interface).
3. The media rental system keeps track of customers and media (movies and music albums). A customer has a name, address, a plan and two lists (queues). One queue represent the media the customer is interested in receiving and the second one represents the media already received (rented) by the customer. There are two plans a customer can have: UNLIMITED and LIMITED. UNLIMITED allows a customer to receive as many media as they want; LIMITED restricts the media to a default value of 2 (this value can be change via a manager method).  
     
   A movie has a title, a number of copies available and a rating (e.g., "PG"). An album has a title, number of copies available, an artist and the songs that are part of the album.
4. You must define and use at least three classes / interfaces (not including MediaRentalManager) as part of your design. **These clases must support the functionality of the system, otherwise you will not receive any credit.**
5. The database for your system needs to be represented using two ArrayList objects. One ArrayList will represent the customers present in the database; the second will represent the media (movies and albums). This is the ArrayList requirement provided in the **Grading** section.
6. You must define javadoc for the classes you implement. You do not need to provide javadoc comments for the MediaRentalManager class. Notice you do not need to run the javadoc utility.
7. Regarding the searchMedia method: the songs parameter represents a substring (fragment) or the full list of songs associated with the album. If the full list is provided, you can assume commas will be part of the string. **Hint:** you may want to consider using the indexOf method of the String class.
8. Feel free to use Collections.sort to sort your data.
9. Although you are not require to write student tests you are encourage to do so.
10. Not all the details associated with the project can be fully specified in this description. The sooner you start working on the project the sooner you will be able to address any doubts you may have.

## Requirements

* Follow the style guide provided at [Style Guide](http://www.cs.umd.edu/~nelson/classes/resources/javastyleguide/).
* You should not modify the project specifications. For example, do not add extra behavior nor output.

## Academic Integrity

Please make sure you read the academic integrity section of the syllabus so you understand what is permissible in our programming projects. We want to remind you that we check your project against other students' projects and any case of academic dishonesty will be referred to the [Office of Student Conduct](https://osc.umd.edu/).